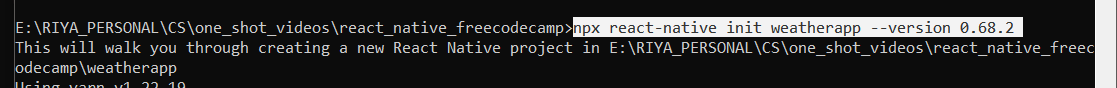
React native:

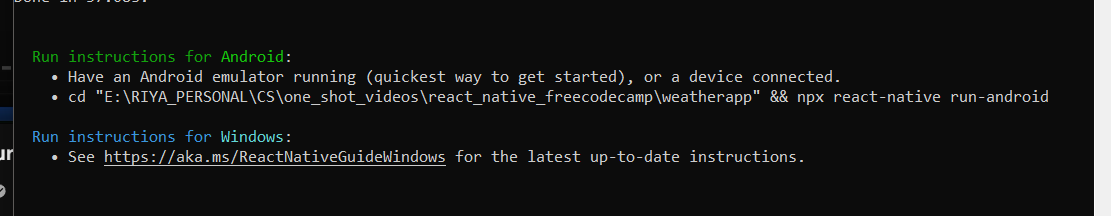
Installation I did:

~~JDK17(java)~~ USE JDK 11 NOT JDK 17 and set JAVA\_HOME environment variable in system env variables, android studio (then downloaded SDK’s and SDK tools and set the environment variable for android SDK in PATH for both user and home), node js (it will also download npm)

Commands to get the application running:

1. To initialize the app and load required files:





1. Get the emulator running
2. Then to start node js, metro run this command in the command prompt after going to the application folder:

A black screen with white text

Description automatically generated

1. If the build id successful we’ll get this message:

A computer screen with white text

Description automatically generated

Then the node js metro command line will open up:

A screenshot of a computer program

Description automatically generated

And the emulator will open the app automatically with the welcome to react native screen:

A screenshot of a video chat

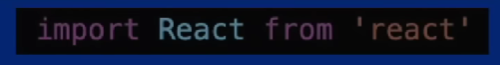
Description automatically generated

React Native basics:

The android and IOS apps will get build automatically as we edit the App.js file. There can also be an App.tsx file which refers to a typescript file, javascript is a subset of typescript so if an App.js file gets built, we can still use JavaScript in it. Typescript offers more functionality than JavaScript.

In the App.js file🡪

We need to import react:

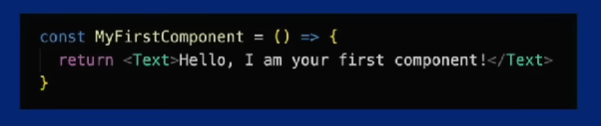


We will also import any of the core components that we need to use:

A black and blue background with text

Description automatically generated

Our component is a constant function so it starts with the keyword “cost” and any component must return a jsx

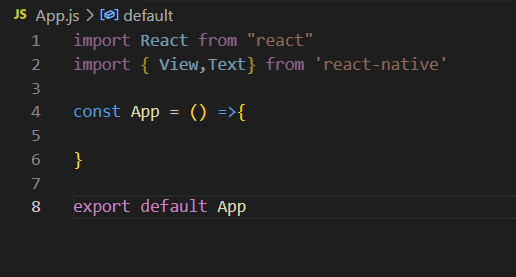


Also, we need to export the functional components.

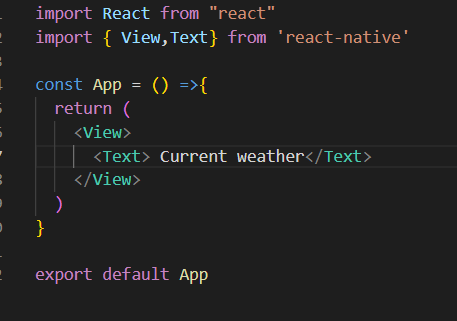
Core component: view, text etc, we import these.

Everything else used in the app is also a component. Elements reside inside the components.

**First basic setup without any UI:**



Returning a basic text to the app:



Now the text will look different according to different emulator or the screen that we use. Adding margins or padding wont work as it will only guarantee the design on the specific emulator that we are working on currently.

To solve this problem we have another core component called the “safe are view”

|  |  |
| --- | --- |
| Without Safe area View: | After using the safe are view:    Iitna kuuch difference lag ni raha tbh |

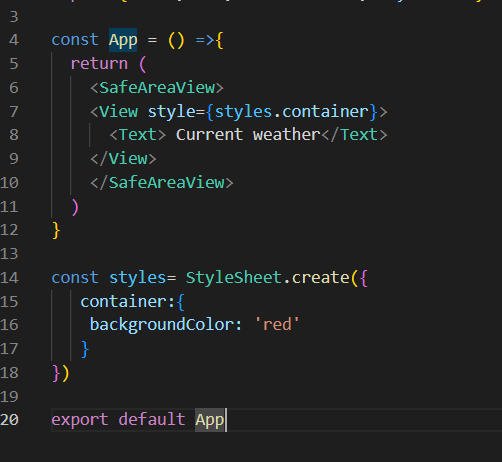
We cant access CSS directly here, so we use a javascript object. Everything is same except we use camel casing:



Inline style are applied to the components/elements as we create them

We can define all our styles in one place called stylesheet.create, we’ll import the stylesheet from reactnative api

Using the stylesheet.create method:



All the components above in the tree should use the same flex as the component which is using the flex.

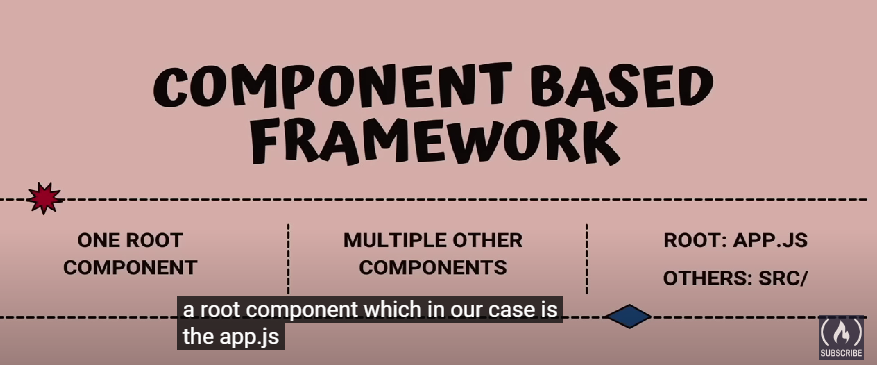
All numbers in react native are unitless and represent density independent pixels, ie absolute measurement.

There are many ways to include icons in the app but we will use a library here called “react native vector icons”, it is on github. The details about the icons can be found on “Icons-expo documentation” as both are the same.

A component returns a jsx and the computer uses babel to convert this jsx into javascript.

We can make custom components and then import them to use them in our application.

React native is a component-based framework.



App.js acts as entry point and multiple other components are stored inside the component’s directory.

Lists:

We have the option of using these core components to use lists in react-native:



A blue and green background with white text

Description automatically generated

**Custom Components:**

This is how custom components/tags are made which can be customized in every use to display something different:



Defined the custom component here.

A screen shot of a computer program

Description automatically generated

Called the custom component with different messages.

**Navigation**

in react native there is no built in way of navigation so we need to use a library to do that, there are 2 main ways to do it:

1. React navigation
2. React native navigation.